Prolog-based framework for privacy evaluation and user feedback

Milica Milutinovic
Bart De Decker

Report CW 688, November 2015

KU Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Prolog-based framework for privacy evaluation and user feedback

Milica Milutinovic
Bart De Decker

Report CW 688, November 2015

Department of Computer Science, KU Leuven

Abstract

In this report, we describe the design and functioning of a logic-based framework for privacy evaluation and user feedback. This framework is meant to be running on the user's side and to track user's actions, i.e. disclosures that she makes, in order to evaluate the profiles different providers hold about her. The evaluation is based on the utilised credential technologies, providers authentication, data storage and data sharing policies and on actual interactions between the user and providers when services are consumed. Besides providing feedback on the achieved privacy level, the framework can also be applied for inspecting the consequences of potential subsequent disclosures. This allows the user to make an informed decision on which disclosures to make or to choose between alternative authentication policies.

Prolog-based framework for privacy evaluation and user feedback

Milica Milutinovic and Bart De Decker

KU Leuven, Dept. of Computer Science, iMinds-DistriNet/SecAnon,
{Milica.Milutinovic, Bart.DeDecker}@cs.kuleuven.be,
https://distrinet.cs.kuleuven.be/

Abstract. In this report, we describe the design and functioning of a logic-based framework for privacy evaluation and user feedback. This framework is meant to be running on the user's side and to track user's actions, i.e. disclosures that she makes, in order to evaluate the profiles different providers hold about her. The evaluation is based on the utilised credential technologies, providers authentication, data storage and data sharing policies and on actual interactions between the user and providers when services are consumed. Besides providing feedback on the achieved privacy level, the framework can also be applied for inspecting the consequences of potential subsequent disclosures. This allows the user to make an informed decision on which disclosures to make or to choose between alternative authentication policies.

1 Introduction

Nowadays, many services are offered as electronic services. As they often provide better customer-experience and efficiency, they are a preferred option of many users. Areas ranging over commerce, finance and governmental services are thus establishing a strong online existence. Most common examples are online shops, social networks, banking sites, or administrative communication with the governmental bodies, such as tax authorities.

While the early versions of online services were offered by a single provider, they are now often provided through a collaboration of multiple entities, each of which covers a specific aspect of the business, using collaboration to increase their efficiency and effectiveness. Every provider in such collaboration has a distinct role, with specific responsibilities and authorisations. In the online shopping example, the shop utilises the order information, while the delivery service requires the user's address and a third party payment service receives the banking data. Additionally, some providers have a business model that does not require the users to pay for a service, but the gathered data about them is monetised by being sold to other parties, such as advertisement providers. Consequently, not only are the users in direct communication with a growing number of online providers, but their personal data is reaching even more entities.

All this represents a challenge for tracking the knowledge databases of providers and assessing an individual's dynamics privacy level. In order to determine the

partial identities that several providers store about a user, it would be necessary to keep track of all user activities, i.e. disclosures, and the alternative flows of information, i.e. exchange of data amongst providers.

In order to tackle this, we built a logic-based framework for providing feedback on the user's privacy level. Our framework is implemented in the Prolog logic programming language and is intended to operate on the user's side and model the observed system and all the disclosures that a user has made, allowing to pose queries about the resulting privacy level, i.e. the knowledge of the providers with which the user directly or indirectly interacts. Besides being beneficial for the users, the framework can also be used by system developers. They would model a chosen system design and specify scenarios of interest, on which queries can be made. This can provide insight on the attained users' privacy level and allow them to detect if privacy requirements are met.

2 The privacy-feedback framework

The privacy-feedback framework is meant to evaluate the attained privacy level of a user by informing her about her partial identities as seen by the providers. It takes as input the relevant technologies, service providers with which the user interacts, and the user's previous actions, i.e. disclosures. The system also takes into account the potential merges of the databases of providers based on their collaborations. As a result, it provides the information on the partial identities as seen by the interacting providers.

Besides getting feedback on the current privacy level, the users are able to make queries about the effect potential disclosures would have on their privacy and help choose between service providers or alternative authentication policies.

The rest of this section describes the framework design, by explaining its conceptual model and the semantic data representations.

2.1 Conceptual model

The conceptual model of the framework is presented in Figure 1. It takes as input a model of the system, which is translated into a readily-executed object code. The user poses queries on this code, the results of which provide information about her privacy level.

The process components are described in the rest of this section.

Input models The input models represent models of the different system aspects. They include the following:

- The *credential model* represents users' credentials used in interactions with providers. For instance, the utilised types of credentials can be national identity cards, loyalty credentials or student cards. This model includes the credential technologies present in the system and credential templates used. The credential technologies are relevant for representing the consequences

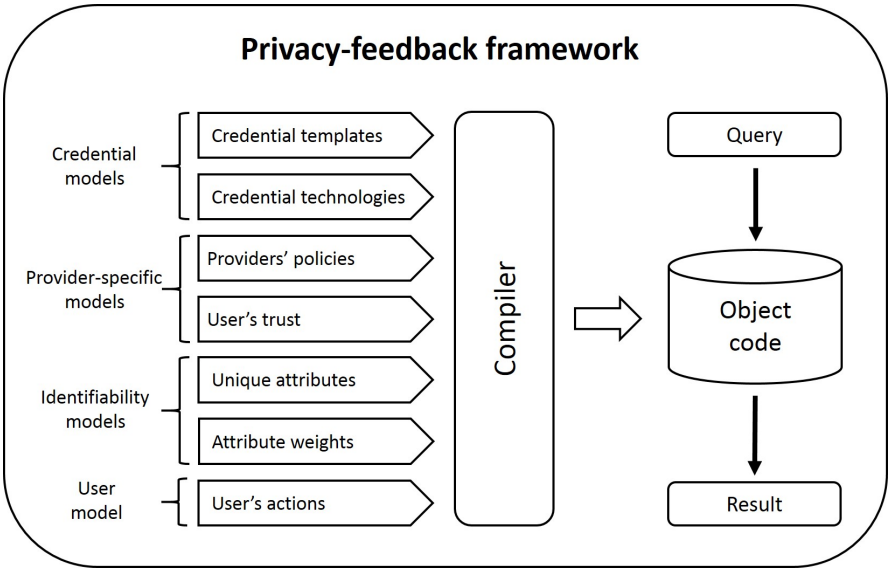

Fig. 1. The conceptual model of the privacy-feedback framework.

of their usage for disclosures. For instance, showing one attribute with an X.509 credential would lead to disclosure of all credential attributes. On the other hand, anonymous credential technologies allow for selective disclosure of attributes. A credential of a specific type and technology is assumed to have a unique structure. The credential templates specify this structure, by defining the set and ordering of attributes in a credential which is defined by a specific technology-type combination.

- The *provider-specific model* defines the data handling practices in terms of collection, storage and possible sharing with other entities. It is based on the privacy policies of providers, that specify which data is collected, processed, stored and made available to other entities. On the other hand, the provider-specific model also specifies users' trust in providers. Namely, if a user trusts the providers' specified privacy policies, the privacy level feedback is constructed based on them. However, if the user does not trust the providers to limit the storage of collected data, or not to share her data with other providers, the model reflects this and provides feedback based on the user's expectations. The feedback can also be constructed for the worst case scenario, where all collected data is considered to be stored, processed and shared with all other observed providers.
- Another input element is the definition of *identifiability models*. On the one hand, it specifies the unique attributes and attribute sets. The uniqueness property is assumed to depend on the source and certifier of an attribute. This information is used for determining existence of links between different user profiles, i.e. determining that they pertain to the same individual.

Two profiles containing the same unique attribute or attribute set are considered to be linked together. The second part of the identifiability model assigns all attributes with a weight. The weight represents the sensitivity of the attribute's disclosure and depends on the kind of attribute and its source, i.e. trustworthiness. For instance, attributes certified by the government are more trustworthy than a free user input, and can thus have a higher weight. This information is relevant for calculating the privacy cost of a disclosure, e.g. by summing up the weights of disclosed attributes or applying more complex algorithms. With this model, the framework can inform the users about quantified privacy costs of specific disclosures, thus helping the disclosure-decision making process.
- Finally, concrete *user actions* are modelled as the final and cumulative part of the input. They represent tracked user behaviour and disclosures towards different providers. Based on them, the current privacy level of a user is determined.

The input models are flexible and can be removed, added and updated as needed. This allows to model different systems and provide an up-to-date view on user privacy.

Queries and results For the production of the feedback, queries are posed on the compiled Prolog model. Posing specific queries allows to obtain information about different aspects of the user privacy level. Examples are:

- Getting an overview of all of the profiles held by chosen or all modelled providers.
- Information on all identifiable profiles.
- Verifying if one of the recorded profiles contains a specific attribute or a set of attributes. This may refer to a fully specified attribute or only an attribute name or value.
- The links between the profiles, their holders and the reasons for each of the links' creations.
- The cost of a particular disclosure.

All of the above queries can also be made for a hypothetical scenario. Namely, the user can define a set of assumptions, including potential actions or providers' trustworthiness, for which she wishes to inspect the achieved privacy level.

Implementation The framework is built using the Prolog declarative logic programming language[1]. It allowed us to model the privacy aspects of a system through the following means:

- Asserted facts create the initial knowledge database of the system. They are also used to specify the collaboration between providers and to define the attribute sets that are considered unique. When determining the privacy cost of a disclosure, they are used to assign weights to specific attributes.

[1] The code is available at `https://github.com/mmilica`

– Rules allow to model the possible information flow in the system. A certain level of 'imperative' programming may be achieved, by invoking a desired action, such as assertion, when a goal or query is executed.
– Data structures are represented with predicates with appropriate number of arguments.
– Queries allow to ask for the list of information held by the providers, list of links that can be made, whether a specific data set is present in a provider's knowledge database, and the expense of a particular disclosure.

2.2 Semantic data representations

The framework handles information in the form of adequate data structures. They capture the information and its relevant metadata. Examples of such structures are user profiles, attributes, attributes' definitions, credential templates and instantiated credentials.

User profiles The data that is disclosed to the providers is modelled with *user profiles*. A profile represents a collection of pieces of information which are revealed by the user within one uninterrupted interaction with a provider, i.e. in one session, and that are stored by the provider[2]. The profiles are created dynamically, as the occurring user actions are modelled. For instance, showing an attribute is modelled as invoking actions of recording the attribute and its metadata in the provider's database and creating applicable links within the providers database and between collaborating providers' databases. Every link between profiles represents the fact that they pertain to the same individual. The way they are formed is presented in Section 3.

Besides consisting of the disclosed information, every profile is identified with a *transaction identifier* (TID). The identifier is freshly generated for each established connection, even with the same user[3]. The identifier is consequently linked to all the disclosed pieces of information within one interaction, including credential disclosures and free user input:

$$Profile_i = \{ TID_i : \{ Attribute_{ij} \}_{j \in \{1..n\}} \} .$$

An illustration of the framework's database is provided in Figure 2. Every service provider SP_k is linked with the profiles it had constructed, identified with TID_{ki}. Every profile contains a set of attributes, each linked with its source information. The source information can point to free user input or a certified

[2] The information storing policy of a provider is also represented with the system model. A disclosed piece of information is modelled as stored if the provider policy specifies so, or if the user does not trust the provider and assumes that all collected data is also stored.

[3] The provider will only be able to deduce the link between different interactions with the same user, if she discloses the same unique attribute or attribute set in both. As the system can model the metadata, such as IP addresses, this can also be used for establishing an existence of a link between profiles.

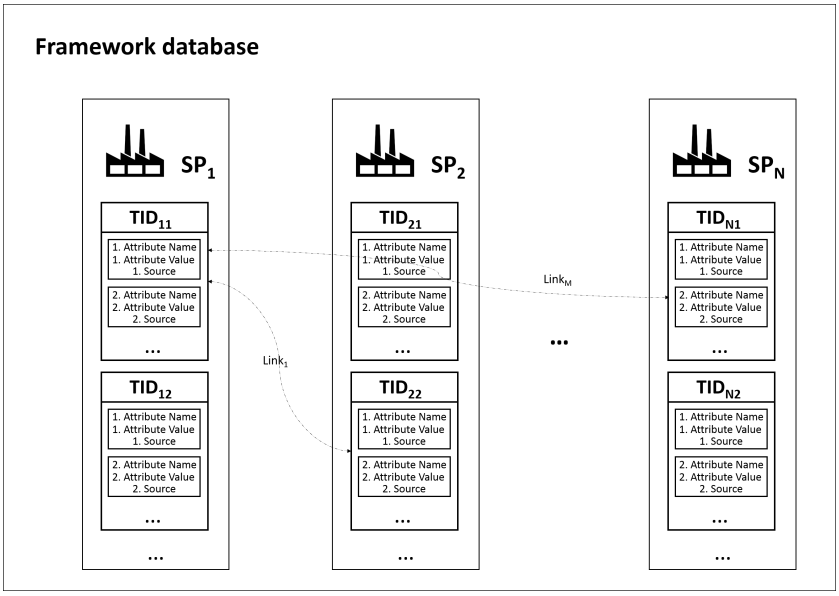

Fig. 2. The knowledge database of the framework.

credential identified by its type, technology and the issuer. The established links between these profiles are denoted with dotted lines.

Every existing profile entry is modelled in Prolog as an asserted fact. The predicate of the fact has as arguments the identifier of the service provider which holds the knowledge of the profile entry, the transaction identifier and a recorded attribute with its source:

$$\texttt{profile}(\texttt{SP},\ \texttt{TID}_\texttt{i},\ \texttt{Attribute}_\texttt{ij},\ \texttt{Source}_\texttt{ij}).$$

In case one provider offers multiple services to users, this is modelled as the existence of multiple providers which collaborate, i.e. share their knowledge databases. The framework considers all the data to be exchanged between them, reflecting the fact that all the data is held by one entity. At the same time, the information about which service was requested by the user is maintained – it is encoded in the service-specific identifier of the interacting provider.

Credentials Two types of credential structures are handled by the system: credential templates and issued credentials.

Credential templates specify the structure of a specific kind of credentials and are used to specify the ordering of attributes they record and their properties (e.g. whether the issuer learns the attribute value and who chooses the value, user or issuer). They are used for determining the position of an attribute of interest within a credential and possibly checking some of its properties, e.g.

uniqueness. Templates do not contain attributes, as values are not assigned yet, but only contain attribute definitions:

$$CredTemplate_i = \{Technology_i, Type_i, [List\ of\ \ AttrDefinitions]_i\}\,.$$

The *Technology* argument refers to the specification of the credential technology, such as X.509 or Idemix, while the *Type* represents its purpose, such as Belgian eID or a store's loyalty credential. Every credential template is uniquely identified with the type and technology combination.

On the other hand, *issued credentials* contain the actual attribute values, but not the properties, as this information is available in the template specifications. Issuance of credentials is modelled as creation of a credential structure with initialised attributes, i.e. attributes with assigned values. It also asserts the ownership of the user interacting with the issuer[4]. However, this does not require the issuer to learn the identity of the user.

Attributes There are two attribute structures used in the framework. One models the attributes' definitions which appear in the credential templates, while the other represents the instantiated credential- and profile-attributes.

The *attributes of a user*, which constitute her (partial) identity, are represented with the attribute name and value:

$$Attribute_{ij}^{Profile_i} = \{AttrName_j, AttrValue_j\}\,.$$

This structure is recorded as part of a user profile or an issued credential. In a user profile, every attribute is paired with its source information.

Attribute definition structure contains the following fields:

$$AttrDefinition_i = \{AttrName_i, ValueType_i, IssuerKnowledge_i, ValueChoice_i, Sharing_i\}\,.$$

The *AttrName* of the attribute does not need to be unique. The *ValueType* defines the type of this attribute's value, such as 'String' or 'Integer'. The *IssuerKnowledge* and *ValueChoice* are fields representing whether the attribute is disclosed to the issuer of the credential and whether the value is assigned by the issuer or chosen by the user, respectively. Finally, the *Sharing* property describes the level of attribute uniqueness. An attribute value can be unique or shared amongst a differing number of users. For instance, a social security number is defined as unique, while name or date of birth are not and address attribute is considered to have a restricted sharing property.

Examples of an attribute structure and an attribute definition (linked to a specific credential) as represented in Prolog are:

[4] This allows the framework to go through the list of user's credentials, e.g. if it is extended to offer determining the optimal choice for satisfying an authentication policy (minimising the cost of the disclosure, as in Section 4.3).

```
attribute('ZIP', '3000').
attrDef('ZIP', 'ZipCode', 'Known', 'Assigned', 'Shared').
```

The definition specifies that the attribute is learned by the issuer of the credential (i.e. 'Known'), it is assigned by the issuer (i.e. 'Assigned') and its value is shared by multiple users (i.e. 'Shared').

Since the framework also models the value aspect of attributes, this allows for extension of the framework's functionality. Namely, some authentication policies require only proving properties of attributes, e.g. age is over 25. For trustworthy providers, it is assumed that only this property is recorded. In addition, the Idemix credential technology supports proving certain properties of attributes, without disclosing their values. Therefore, the framework can be extended to model collection or storage of only attribute properties.

3 Feedback creation

When a user reveals a piece of information, it is recorded by the interacting provider as a profile entry[5]. After the new entry is created, the framework tries to create links to other recorded profiles. It first checks whether the attribute is unique in which case the same attribute is searched for in other profiles. If a match is found in other profiles, they are marked as linked. Every link between profiles is recorded through a linking predicate with profiles' transaction identifiers as arguments. It also specifies the reason for linking, i.e. the unique attribute. Next, the unique attribute sets are evaluated. Similarly as with unique attributes, the links are recorded with the set definition as a reason for creating the link. In Prolog, a user action, e.g. a disclosure is represented as a head of a rule, the body of which includes performing the needed checks, e.g. which type of credential is used and which consequences it entails, and invoking profile entries creation.

Initially, the user requests a specific service (`reqService`) from a service provider (`SP`) over a transaction identified with `TID`. The related Prolog rule has the form[6]:

```
reqService(User, SP, TID, Service) :-
    authPolicy(SP, Service, ListOfAttrAndSourcePairs),
    authenticate(User, SP, TID, ListOfAttrAndSourcePairs),
    showMetadata(User, SP, TID, MetaData).
```

[5] Whether a piece of information is stored depends on the provider's storage policy. Also, if the user does not trust the provider to limit the storage, all collected data is considered to be stored by the provider.

[6] The presentation of the implemented Prolog rules is optimised for clarity. The actual implementation of the presented examples may be more compact or more detailed and differs slightly in the used notations.

Upon requesting a service, the authentication policy of the provider is consulted (`authPolicy`). It specifies which attributes the user shows in the interaction and their sources, e.g. specific credentials or free user input (`ListOfAttrAndSourcePairs`). Once the applicable policy is determined and the list of required disclosures, the authentication is modelled. If there is network level data that the provider obtains, which could be used for deriving links between user actions, it is also included as part of the disclosure (`showMetadata`). For instance, if a user utilises a static IP address for multiple sessions with providers, it is recorded as part of the created profiles, as it can also be used for creating links between profiles. The disclosures from different sources (e.g. credential source and free user input) are modelled differently. In case of credential source the applicable rule is:

```
authenticate(User, SP, TID, [First|Rest]) :-
    First = attrAndSource(cred(_, Technology, Type, _), AttrList),
    Cred = cred(User, Technology, Type, _),
    showCred(User, SP, TID, Cred, AttrList),
    authenticate(User, SP, TID, Rest).
```

In this case, a credential of the user that has the required type and technology is found and its show is modelled. If the disclosure refers to user input, the applicable rule is:

```
authenticate(User, SP, TID, [First|Rest]) :-
    First = attrAndSource('userInput', AttrList),
    userInput(User, SP, TID, AttrList),
    authenticate(User, SP, TID, Rest).
```

If the user discloses information by showing a credential, the utilised credential technology has an impact on the resulting disclosure. If the disclosure of a piece of information is performed by having the user show an X.509 credential attribute, the applicable Prolog rule has the form:

```
showCred(User, SP, TID, Credential, AttributeList) :-
    Credential = cred(Alias, Technology, Type, ListOfCredAttrs),
    Type = 'X.509',
    createFullListOfAttributesToShow(List, ListOfCredAttrs),
    showCredentialAttributes(User, SP, TID, Credential, List).
```

This rule models the user showing a specific provider (`SP`) a chosen list of credential attributes (`AttributeList`) over a transaction identified with `TID`. The user's credentials are also represented with aliases, which are used to assist the user-side credential management. For showing an X.509 credential, the list of attributes to be disclosed to the provider is created as a full list of credential attributes (`createFullListOfAttributesToShow`). In case of Idemix credentials,

only the specified attributes are shown (`createListOfChosenAttributesToShow`), as selective disclosure is supported. In addition, it is also possible to only prove ownership of an anonymous credential. The rule applicable to this case has the form:

```
showCred(User, SP, TID, Credential, AttributeList) :−
    Credential = cred(Alias, Technology, Type, ListOfCredAttrs),
    Type = 'Idemix',
    createListOfChosenAttributesToShow(List, AttributeList),
    showCredentialAttributes(User, SP, TID, Credential, List).
```

Showing a list of attributes is a recursive rule, which adds the attributes to the provider's knowledge database one by one, if specified in the storage policy. The credential source is also listed as part of the database entry. Additionally, it adds the entry to the collaborating providers' databases, in case the sharing policy specifies exchange of the given attribute:

```
showCredentialAttributes(User, SP, TID, Credential, List) :−
    List = [AttributeNameAndValue|Rest],
    Credential = cred(Alias, Technology, Type, ListOfCredAttrs),
    getCredAttribute(Credential, 'Issuer', Issuer),
    Source = credSource(Technology, Type, Issuer),
    addToProvidersDB(SP, TID, AttributeNameAndValue, Source),
    addToCollaboratingProvidersDB(SP, TID, AttributeNameAndValue,
    Source),
    showCredentialAttributes(User, SP, TID, Credential, Rest).
```

An attribute that is disclosed to the provider is recoded in the knowledge database either if the worse case is observed, or the user does not trust the provider or the provider's storage policy specifies storage of this attribute. Making an entry in the providers database invokes the following actions. Firstly, a check whether the profile entry exists in the database is performed. In case there is no matching entry, the entry is asserted as part of the provider's knowledge database. Secondly, the creation of links between other profile entries, i.e. other profiles identified with differing transaction identifiers, is initiated. Adding an attribute to a collaborating provider's database is performed in a similar way. It only requires consulting the information sharing policy of the original provider

to check whether the specific attribute is shared with the collaborating provider.

```
addToProvidersDB(SP, TID, Attribute, Source) :-
    (
        isWorstCase
        ; noUserTrust(SP)
        ; isStoredBySP(SP, Attribute, Source)
    ),
    (
        profileEntry(SP, TID, Attribute, Source), !
        ; assert(profileEntry(SP, TID, Attribute, Source)
    ),
    establishLinksBetweenProfiles(SP, TID, Attribute, Source).
```

The links establishment starts with a check whether the given attribute, from the specified source, is unique. In the case it is, the attribute is searched for in other profiles. If a match is found, a link between the profiles is asserted with a specification of the reason for its creation (e.g. matching unique citizen number from the national ID card), unless the link was already recorded. If the link already exists, but was recorded for another reason, the list of linking reasons is simply extended. The link extends to other profiles linked to the one with which a link is being established. The process is repeated for all matches that are found:

```
establishLinksBetweenProfiles(SP, TID, Attribute, Source) :-
    isUnique(Attribute, Source),
    Attribute = attribute(Name, Value),
    transactioID(TIDOther),
    \+ TID = TIDOther,
    searchForUniqueAttrInOtherProfiles(SP, TIDOther, Attribute,
    Source),
    recordProfilesLink(SP, TID, TIDOther,reason(uniqueAttr(Name,
    Source))),
    fail.
```

If the observed attribute is not unique, the sets of unique attributes which it constitutes are searched for next. After determining the unique sets this attribute belongs to, the sets are searched for in the provider's database. The search examines whether a set exists in the profile which the attribute belongs to and all its linked profiles. If the provider records the unique set in a single or across linked profiles, they are then searched for in other (linked) profiles. In case of a

match, the profiles are denoted as linked. The process is repeated for all (linked) profiles that contain the examined unique attribute set. The process is repeated for all unique sets that contain the observed attribute. A simple illustration of this rule is:

```
establishLinksBetweenProfiles(SP, TID, Attribute, Source) : −
    searchForUniqueSetInLinkedProfiles(SP, TID, Attribute, UniqueSet),
    transactioID(TIDOther),
    \+ TID = TIDOther,
    searchForUniqueSetInOtherProfiles(SP, TIDOther, UniqueSet),
    Reason = reason('UniqueSetInLinkedProfiles', UniqueSet),
    recordProfilesLink(SP, TID, TIDOther, Reason), fail.
```

A few examples of links recorded by providers are illustrated in Table 1. The examples illustrates the assumption that the *public key* a user has in her student card is a unique attribute and that set of *address* and *date of birth* from a Belgian national identity card attributes form a unique attribute set.

Table 1. Examples of recorded links between user profiles.

Service provider	**Linked profiles**	**Reason of link creation**
Provider A	TID_i & TID_j	uniqueAttribute{'PK' in ('X.509', 'student card', 'University')}
Provider B	TID_k & TID_l	uniqueAttrSet{'Address' in ('X.509', 'ID card', 'Government'), 'DoB' in ('X.509', 'BeID', 'Government')}

For establishing the links, it is important to know which service providers collaborate in the sense that their recorded databases are exchanged. Every service provider has a policy describing the authentication and disclosure requirements placed on the users interested in the offered services. In addition, data management policies state which obtained data is stored and how the data is further distributed. However, the providers are not necessarily trusted to handle the obtained data in the specified way. User trust can be divided into two categories:

- Users trust the service provider's information storage and distribution policies, meaning that the storage of data and collaborations between providers is modelled as stated in their privacy policies.
- The providers are not trusted to limit the storing of collected data or the distribution of data to entities as stated in their policies. In this case, the users may assume that the provider stores more or all data that it collects.

Similarly, it may be assumed that certain providers share their databases, or parts of their databases. The links between user profiles are established according to the user's assumptions, i.e. the level of trust in the providers.

From the point of view of finding new links between profiles, multiple profiles are observed as one if there is an existing link between them. This means that with every link that is created, the possibility of creating new links usually increases. This is because a larger number of linked profiles has a larger set of recorded attributes observed as belonging to one profiles. This increases the probability of having an additional unique set that can be used for further links creation.

The fact that the user attributes are modelled with their actual value allows to enforce link creation based on the values, even if the source information differs. Posing restrictions on trusted sources which allow for creating links is also possible.

After the user's actions are modelled, the knowledge database can be examined through queries. An example of a query is checking which profiles of the user are created. After posing the query:

$$? - \texttt{profile}(\texttt{SP},\ \texttt{TID},\ \texttt{Attribute},\ \texttt{Source}).$$

The answer will consist of multiple elements, part of which is:

$\texttt{SP} = \ '\texttt{AdsProvider}',$
$\texttt{TID} = \ '\texttt{99}',$
$\texttt{Attribute} = \texttt{attribute}('\texttt{StudentNumber}',\ '\texttt{12345678}'),$
$\texttt{Source} = \texttt{credentialSource}('\texttt{X.509}',\ '\texttt{studentcard}',\ '\texttt{KULeuven}'),$

$\texttt{SP} = \ '\texttt{AdsProvider}',$
$\texttt{TID} = \ '\texttt{99}',$
$\texttt{Attribute} = \texttt{attribute}('\texttt{DoB}',\ '\texttt{01.01.1980}'),$
$\texttt{Source} = \texttt{credentialSource}('\texttt{X.509}',\ '\texttt{BeID}',\ \texttt{government}),$

The user can also query all the information about her that a specific provider holds, or all profiles that contain a specific attribute:

$$? - \texttt{profile}('\texttt{OnlineTravelAgency}',\ \texttt{TID},\ \texttt{Attribute},\ \texttt{Source}).$$

Similarly, the user can query which links are derived between her profiles:

$$? - \texttt{isLinked}(\texttt{SP},\ \texttt{TID1},\ \texttt{TID2},\ \texttt{Reason}).$$

The result would be in the form:

$$\begin{aligned}&\texttt{SP} = \texttt{'AdsProvider'},\\&\texttt{TID1} = \texttt{'99'},\\&\texttt{TID2} = \texttt{'201'}\\&\texttt{Reason} = \texttt{uniqueAttribute}(\texttt{'PublicKey'}, \texttt{'BeID'}, \texttt{'government'}),\end{aligned}$$

The framework feedback relies on two types of identifiability quantification. As considered previously, the first provides the definition of (sets of) attributes from specific sources that allow to pinpoint an individual with some assurance. The second model is the definition of the weights assigned to concrete user attributes. They represent the cost of disclosure of the specific attribute. This information is considered to be dynamic and for optimal results, it would be provided by a trusted party which is assumed to have better overview on user attributes' statistics, but also be familiar with the potentials of the data mining techniques. For increased preciseness, the attribute weight would also depend on the actual attribute value.

The information about a cost of a disclosure can also be obtained. By querying for a cost, the verification of a head of the applicable rule invokes the calculation, i.e. summation of the attribute weights and the result is printed for the user:

$$?- \texttt{createWeightCount}(\texttt{User}, \texttt{SP}, \texttt{TID}, \texttt{ListOfCredsAndRevealedAttrsPairs}).$$

4 Use cases

In order to illustrate the use of the framework, three simple use cases are considered. The first illustrates usage of the framework for getting feedback on the current privacy level of the user. The second and third demonstrate how the framework can assist the user with making disclosure decisions and with choosing between different disclosure policies of a provider, respectively[7].

In the examples, the user is assumed to own a set of credentials. The issued credentials contain the 'expiration' attribute, limiting their validity. This attribute is used in every interaction to prove credential validity, but is not modelled in this work for simplicity reasons. We assume that its value is created in a way that does not increase the chances of linking different interactions in which the same credential is used. Additionally, the issuers of the users' credentials learns certain attributes of the users at the time of issuance. The framework can model the issuers as providers that store their profiles created based on

[7] The code that implements the presented use cases can be found at `https://github.com/mmilica`

the issuance interaction. However, for simplicity reasons, the issuers' knowledge databases are not modelled in the presented use cases.

Some of the credentials that the user is assumed to own are of type X.509. This standard specifies that the credential contains, among other attributes, a serial number and the owner's public key. The serial number is unique for every certification authority and thus allows to uniquely identifying a credential. The public key is also a uniquely identifying attribute. In the observed examples, we do not model these attributes for credentials that contain another unique attribute, as one unique attribute suffices to uniquely identify the credential and allow for linking of different sessions where the credential is presented.

The user is assumed to own the credentials represented in Table 2:

- A government-issued electronic ID card, containing the following attributes[8]:
  AttrList1 = {Name, Surname, DoB, Address, PK}
- An electronic credential representing a shopper card. This can be a credential linked to the loyalty scheme that a shop offers. It is issued by the online shop and contains the following attributes:
  AttrList2 = {ZIP Code, Customer number}
- A student card issued by a university that records:
  AttrList3 = {Name, Surname, Student number, University, PK_S}
- An anonymous credential containing the same attributes as the electronic identity card, recertified by a trusted third party (TTP).

The system's identifiability models assume the following:

- All modelled public keys have unique values.
- The customer number contained in a shopper card of a specific provider is assumed to be unique.
- It is assumed that student numbers are unique per university which issues them.
- The issuance of student cards is assumed to allow for unique identification of an individual through the combination of the student number and the date of birth attributes.

Table 2. User's credentials.

Credential	**Type**	**Tech.**	**Attributes**	**Issuer**
eID	Identity card	X.509	*AttrList1*	Government
Shopper card	Shop-issued card	X.509	*AttrList2*	Online shop
Student card	Student status card	X.509	*AttrList3*	University
ID card	Identity credential	Idemix	*AttrList1*	TTP

[8] 'DoB' in the list of attributes represents a date of birth and 'PK' denotes a public key.

In the observed examples, the providers have specific authentication policies, but are assumed to collect all data that is disclosed to them.

4.1 Feedback on user privacy level

The system being modelled is an online store, with which a user engages in multiple interactions over time. The store's authentication policy requires the user to prove the age of majority using a government-issued credential in order to receive the services[9]. Optionally, the user may also provide proof of a student status, for obtaining a discount. For this, a university-signed student card is accepted. However, the user has low trust in the store and believes it stores all data it receives in an interaction, not just the data specified in the authentication policy.

The online store is further assumed to collaborate with an ads provider, which subsequently sends personalised ads to the users through the store's site. The ads provider also collaborates with other service providers with which the user is assumed to have previously interacted. Therefore, before an initial purchase at the observed shop, the ads provider already holds a profile of this user. It is assumed to contain the user's student number and date of birth, but also information on a previous purchase. More details on the actual implementation of this use case are provided in Appendix Appendix A.

A relevant snippet of the providers' databases is presented in Table 3. The links are also established and the existing links are based on the student number that is revealed.

Table 3. The initial state of the knowledge database.

Service provider	**Recorded profiles**	**Established links**
Online shop	/	/
Ads provider	TID_1 : {Student number_U, DoB_U, Purchases_1}	/

In the interactions with the online shop (Table 4), the user makes purchases 'Purchase2' and 'Purchase3' on two separate occasions, disclosing the required date of birth from the identity card and the student status using her student card.

The resulting state of the providers' knowledge databases after the two interactions is represented in Table 5[10]. The online shop records two profiles. However, they contain all the attributes in both the electronic identity card issued by the

[9] We do not model the disclosure of the bank details for payment purposes, and assume that the user remains anonymous through the payment procedure.

[10] The actual attribute values are also recorded in the database, but are not listed in the table for clarity reasons.

Table 4. The state of the updated knowledge database (a).

Service provider	**Transaction ID**	**Used credential**	**Disclosed attributes**
Online shop	TID_2	BeID	$\{DoB_U\}_{BeID}$
		Student card	$\{Ownership\}_{UniCard}$
		Free input	$\{Purchase_2\}_{Free}$
Online shop	TID_3	BeID	$\{DoB_U\}_{BeID}$
		Student card	$\{Ownership\}_{UniCard}$
		Free input	$\{Purchase_3\}_{Free}$

government and the student card. This is the consequence of using X.509 credentials with providers which are assumed to store all received data. The two profiles are also linked as belonging to the same user, as they contain unique attributes, such as public keys and the unique set consisting of the student number attribute in combination with the university or the date of birth. The two purchases are thus concluded to be done by the same individual and the reasons for their linking are recorded as:

1. uniqueAttribute{'PK' in ('X.509', 'student card', 'IssuingUniversity')}
2. uniqueAttribute{'PK' in ('X.509', 'BeID', 'Government')}
3. uniqueAttrSet{'StudentNumber' in ('X.509', 'student card', 'IssuingUniversity'), 'University' in ('X.509', 'student card', 'IssuingUniversity')}
4. uniqueAttrSet{'StudentNumber' in ('X.509', 'student card', 'IssuingUniversity'), 'DoB' in ('X.509', 'BeID', 'Government')}

The ads provider updates its database with the same two profiles, but is able to make even more links and to associate earlier purchases made with other providers. At this point, it can link the users identity to the earlier (pseudonymous) purchase. The reason for creating these additional links is also recorded:

1. uniqueAttrSet{'StudentNumber' in ('X.509', 'student card', 'IssuingUniversity'), 'DoB' in ('X.509', 'BeID', 'Government')}

4.2 Choosing between service providers

If a user is presented with a choice of different service providers offering the same service, but having different authentication policies, the framework can be used to determine which of them offers the preferred privacy level to the user. For illustrating this use case, we use an example of two online bookshops. Their policies are represented in Table 6.

Both bookshops offer their services only to registered users and impose a minimal age requirement. The first bookshop requires the user to authenticate by disclosing the age recorded in her identity card issued by the government and

Table 5. The state of the updated knowledge database (b).

Service provider	**Recorded profiles**	**Established links**
Online shop	TID_2 : {Student number_U, $\text{University}_\text{U}$, Name_U, Surname_U, $\text{PK}_{\text{S}-\text{U}}\}_{UniCard}$, {$\text{Name}_\text{U}$, Surname_U, Address_U, DoB_U, $\text{PK}_\text{U}\}_{BeID}$, $\{\text{Purchases}_2\}_{Free}$	{TID_2, TID_3}
	TID_3 : {Student number_U, $\text{University}_\text{U}$, Name_U, Surname_U, $\text{PK}_{\text{S}-\text{U}}\}_{UniCard}$, {$\text{Name}_\text{U}$, Surname_U, Address_U, DoB_U, $\text{PK}_\text{U}\}_{BeID}$, $\{\text{Purchases}_3\}_{Free}$	
Ads provider	TID_1 : {Student $\text{number}_\text{U}\}_{UniCard}$, $\{\text{DoB}_\text{U}\}_{BeID}$, $\{\text{Purchase}_1\}_{Free}$	{TID_1, TID_2, TID_3}
	TID_2 : {Student number_U, $\text{University}_\text{U}$, Name_U, Surname_U, $\text{PK}_{\text{S}-\text{U}}\}_{UniCard}$, {$\text{Name}_\text{U}$, Surname_U, Address_U, DoB_U, $\text{PK}_\text{U}\}_{BeID}$, $\{\text{Purchases}_2\}_{Free}$	
	TID_3 : {Student number_U, $\text{University}_\text{U}$, Name_U, Surname_U, $\text{PK}_{\text{S}-\text{U}}\}_{UniCard}$, {$\text{Name}_\text{U}$, Surname_U, Address_U, DoB_U $\text{PK}_\text{U}\}_{BeID}$, $\{\text{Purchases}_3\}_{Free}$	

Table 6. Service providers and their authentication policies.

Service provider	**Authentication policy**
Bookshop A	eID_gov: {DoB} Shopper card: {Customer number} User input: Shipping address
Bookshop B	eID_gov or ID_TTP: {DoB} Shopper card: {Customer number} User input: Shipping address

Table 7. Service providers and their storage policies.

Service provider	**Authentication policy**
Bookshop A	Shopper card: {Customer number} User input: Shipping address
Bookshop B	eID_gov or ID_TTP: {DoB} Shopper card: {Customer number} User input: Shipping address

showing the customer number from a certified customer credential and input the shipping address. On the other hand, the second bookshop allows to prove that the age requirement is fulfilled by using either the government-issued ID card or the identity credential issued by a trusted third party. Their data storage policies are represented in Table 7.

Untrusted providers. We first observe the case where providers are not trusted to limit the storage of the collected data. The profiles that each of the providers composes after the user authentication are represented in Table 8. It is clear that even though the same attributes are required to be shown by both providers, the second provider offers better privacy to the users. This is because using the government-issued eID reveals all the credential contents, rather than the singe required attribute. Moreover, the unique attributes such as public key[11] allow to create links with other usages of the same certificate. On the other hand, the anonymous credential issued by the trusted third party allows for selective disclosure, thus offering more privacy to the user.

Table 8. The resulting profiles in untrusted providers' databases.

Service provider	**Recorded information**
Bookshop A	$\{\text{Name}_\text{U}, \text{Surname}_\text{U}, \text{Address}_\text{U}, \text{DoB}_\text{U}, \text{PK}_\text{U}\}_{eID}$, $\{\text{Customer number}_\text{U}, \text{ZIP code}_\text{U}\}_{ShopperCard}$, $\{\text{Shipping address}_\text{U}\}_{Free}$
Bookshop B	$\{\text{DoB}_\text{U}\}_{IDttp}$, $\{\text{Customer number}_\text{U}, \text{ZIP code}_\text{U}\}_{ShopperCard}$, $\{\text{Shipping address}_\text{U}\}_{Free}$

Trusted providers. In case the providers are trusted to store the data according to their policies, the resulting states of their knowledge databases after interaction with the user are represented in Table 9. Even though Bookshop B was determined to have a more privacy-friendly policy when all the collected data is stored by providers, as it allows usage of credential technologies which support selective disclosure of attributes, this example shows that for trusted providers the result can differ. Namely, as Bookshop A does not store the date of birth of the user, it offers better resulting privacy level to the user. Therefore, even though X.509 technology is used for authentication with this provider, it is still a preferable choice due to its more privacy-preserving storage policy.

4.3 Choosing between authentication policies

The framework can also support a more complex feedback provision about user privacy level dependent on the sensitivity of particular attributes. If a user is

[11] A unique attribute is also the certificate's serial number, which is not modelled here.

Table 9. The resulting profiles based on (trusted) providers' policies.

Service provider	**Recorded information**
Bookshop A	$\{\text{Customer number}_{\text{U}}\}_{ShopperCard}$, $\{\text{Shipping address}_{\text{U}}\}_{Free}$
Bookshop B	$\{\text{DoB}_{\text{U}}\}_{IDttp}$, $\{\text{Customer number}_{\text{U}}\}_{ShopperCard}$, $\{\text{Shipping address}_{\text{U}}\}_{Free}$

Table 10. Alternative authentication policies.

Alternative disclosures	**Authentication requirements**
Policy A	eID_{gov}: {Name, Surname, DoB}
Policy B	ID_{TTP}: {Name, Surname, DoB}
Policy C	Student card: {University} & ID_{TTP}: {Name, Surname}

presented with alternative disclosure policies, the framework can be used to make a choice between them taking into account the specific attributes which are required and their sources.

Three exemplary authentication alternatives are listed in Table 10. However, the providers are not trusted to limit their storage of collected data to the attributes required for authentication, but are assumed to store all data that is disclosed in an interaction.

One of the framework input models represents the attribute weights for attributes of different credentials or free user input. The higher the sensitivity of an attribute, the greater the associated weight. The weights for the given example are listed in Table 11. These weights can be adjusted according to the uniqueness of specific values of the user's attributes. This reflects the fact that a less frequent attribute value, such as a unique name and surname, can be considered more sensitive.

The resulting cost of a disclosure is calculated as a sum of the disclosures. When the three different authentication requirements are modelled and the possible disclosures evaluated, Policy B is determined to be the least costly, with the total cost of disclosures being 2.5. Policy A and C are related to the total costs of 4.3 and 5.6, respectively. Although the same attributes are required to be disclosed with policies A and B, the framework result shows that their respective costs are not equal. This reflects the fact that using different credential technologies has different effects on user privacy. Namely, The eID credential of type X.509 discloses all recorded attributes, while anonymous credential technology allows for selective disclosure. Since the provider are not trusted to limit their storage, and are assumed to store all the data that is disclosed in an in-

Table 11. Attribute weights.

Credential type	**Attribute**	**Weight**
eID	Name	0.5
	Surname	0.8
	DoB	1.2
	Address	3
	Public key	1.5
ID_{TTP}	Name	0.5
	Surname	0.8
	DoB	1.2
Student card	Name	0.5
	Surname	0.7
	Student number	1.3
	University	0.3
	Public key	1.5

teraction, the anonymous credentials are a more desirable solution and offer a better privacy level.

5 Related work

For quantitative analysis of user privacy, Reiter and Rubin introduce the notion of *degree of anonymity* [14], which is a continuum ranging from absolute privacy to provable exposure. The actual quantification for the networks for anonymous connections is presented by Diaz et al. [9]. The degree of anonymity is dependent on the probability of an entity being the origin of a message. Serjantov et al. [15] also propose an information theoretic measure of anonymity for mix-based anonymity systems. It is also argued that the knowledge of the size of this anonymity set is not sufficient to determine the anonymity level of a single user, unless all users of the set are completely equal from the point of view of identifiability.

As indicated by Clauß and Schiffner [4], it is not sufficient to observe only the communication layer. Even when the users are indistinguishable at this level, application-level data determines the anonymity level of a user. For instance, in an interaction with a provider, a user may disclose a set of personal attributes (at the application level), each of which is not highly sensitive and does not significantly reduce the anonymity set. However, the attributes collectively may suffice to derive the user's identity. In order to evaluate the privacy level of a user, we take multiple factors into consideration. Those are the kinds of attributes that the user reveals in the interactions with a provider, the frequency of the attribute

values[12] and the provider's access to other sources of user's data. Our approach is also compatible with different metrics for quantifying privacy properties (e.g. the framework proposed by Clauß [3] for quantification of anonymity and linkability). The obtained output of the framework can be combined with the chosen metrics to quantify the achieved privacy properties of a user.

For these evaluations, it is important to know which information providers collect, store and share. Even when privacy policies are available to users, they may not provide expected information or may not be easy to understand [13, 11]. Some proposals aim to offer a uniform privacy policy format and make the policies more useful for the users. Examples are P3P [5] and the CARL [2] language. They are, however, rarely utilised by providers. There are also possible policy conflicts when multiple entities collaborate, which might be hard to detect [1]. In the proposed system, the users are enabled to evaluate their privacy in differing scenarios. Firstly, it is possible to take into account available information about providers' data management and sharing practices. Secondly, if providers are not trusted to store a limited amount of information or limit their sharing, the user may inspect her privacy level under the assumption that providers store all collected information and offer them to other providers. Additionally, the framework tackles the problem of users not being sufficiently familiar with the utilised credential technologies to estimate their impact on privacy. For instance, if X.509 credentials are used for authentication, verification of the credential requires access to all of its contents. This results in disclosure of all credential attributes, even if the provider requested only a limited set. Storing the collected data may be limited to only the requested set, but some providers might not be trusted to do so.This is modelled in the framework and automatically taken into account when the privacy-level results are created.

There are other approaches that analyse privacy by modelling knowledge of users' personal information. Veeningen et al. [17] describes a three-layer model for personal information containing abstract and concrete description of the information. Similarly, we take into account the abstract data description, i.e. what the data represents, but also its concrete content. In addition, some metadata is captured, such as the entity who vouches for the validity of the user's information or the transaction in which the information is disclosed and the service which it is disclosed for. The system is able to take this information into account for modelling the provider's deduction mechanism.

Our framework is based on the Prolog logic language. Usage of Prolog as a security analysis tool and a law verifier was previously explored by Ho and Sundaram [10]. It was applied to the HIPAA act [16] to check whether unauthorised entities can obtain access to resources. The findings suggested Prolog to be useful for detecting vulnerabilities, although it required an adequate setup, i.e. some knowledge of what needs to be inspected. We avoid these issues, as we use Prolog to model system behaviour rather than to search for vulnerabilities.

Other logic programming languages were explored for a formal approach of privacy modelling. An example is work of Decroix et al. [6, 7], which is based

[12] This data includes the behavioural data as well as personal attributes.

on the IDP system [12]. The IDP framework allows to model a system by defining its initial state and the rules which describe its functioning. Based on this model, all possible system states are discovered and can be queried. The IDP system is convenient for evaluating a system from the point of view of showing how an undesirable state can be reached, thus demonstrating that the privacy guarantees are defeated. Although IDP system offers an intuitive way to model systems, its main drawback is the complexity constraint. It allows to analyse only systems of limited complexity. In other words, only a certain number of actors and actions can be observed. Another important feature of this related work is that attributes are observed at a conceptual level, i.e. without modelling the actual attribute values. Since differing frequency of an attribute value affects the overall privacy level, we choose to take this aspect into account. By modelling the actual contents of the exchanged attributes, the framework can further be combined with complex algorithms for determining the privacy level [3]. The usage of IDP was also explored by applying it to loyalty systems [8]. For evaluation, we apply our framework to a privacy-preserving public transport ticketing system.

6 Conclusions

With the proliferation of electronic services, an average user is engaging in a growing number of online interactions. In these interactions personal information is disclosed for authentication or service delivery purposes. In order to manage their identity, users need to keep track of the partial identities constructed by the service providers. However, this is not an easy task, due to the complexity of data flows or unclear data handling policies of providers. In addition, different technologies have differing effects on user privacy, which is something users need to be familiar with for attaining control over their identity management.

In order to facilitate the identity management task by allowing users to view their partial identities held by different providers, we have devised a privacy-feedback tool. It is implemented in the Prolog logic programming language and it models the observed system and related data handling to inform the users which information about her is held by which party. The results are dependent on the specific disclosures, credential technologies, and data sharing policies of providers and identifiability conditions which allow for linking different partial identities. In addition, the framework assists the user to make a disclosure decision by reviewing the impact it would have on her privacy level. This can also be used to make a choice between providers with differing privacy policies. Finally, the framework can also provide the cost of a disclosure based on a given model of attribute weights.

References

1. Travis D Breaux and Akhila Rao. Formal analysis of privacy requirements specifications for multi-tier applications. In *21st IEEE International Requirements Engineering Conference (RE)*, pages 14–23. IEEE, 2013.

2. Jan Camenisch, Sebastian Mödersheim, Gregory Neven, Franz-Stefan Preiss, and Dieter Sommer. A card requirements language enabling privacy-preserving access control. In *Proceedings of the 15th ACM symposium on Access control models and technologies*, pages 119–128. ACM, 2010.
3. Sebastian Clauß. A framework for quantification of linkability within a privacy-enhancing identity management system. In *Emerging Trends in Information and Communication Security*, pages 191–205. Springer, 2006.
4. Sebastian Clauß and Stefan Schiffner. Structuring anonymity metrics. In *Proceedings of the second ACM workshop on Digital identity management*, pages 55–62. ACM, 2006.
5. Lorrie Faith Cranor. P3P: Making privacy policies more useful. *IEEE Security & Privacy*, (6):50–55, 2003.
6. Koen Decroix, Jorn Lapon, Bart De Decker, and Vincent Naessens. A formal approach for inspecting privacy and trust in advanced electronic services. In *Engineering Secure Software and Systems*, pages 155–170. Springer, 2013.
7. Koen Decroix, Jorn Lapon, Bart De Decker, and Vincent Naessens. A framework for formal reasoning about privacy properties based on trust relationships in complex electronic services. In *Information Systems Security*, pages 106–120. Springer, 2013.
8. Koen Decroix, Jorn Lapon, Laurens Lemaire, Bart De Decker, and Vincent Naessens. Formal reasoning about privacy and trust in loyalty systems. In *18th International Conference on Business Information Systems*. Springer, 2015.
9. Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In *Privacy Enhancing Technologies*, pages 54–68. Springer, 2003.
10. Anthony Ho and Sharada Sundaram. A Prolog based HIPAA online compliance auditor. *Unpublished class report*, 2008.
11. Carlos Jensen and Colin Potts. Privacy policies as decision-making tools: an evaluation of online privacy notices. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 471–478. ACM, 2004.
12. Maarten Mariën, Johan Wittocx, and Marc Denecker. The IDP framework for declarative problem solving. *Search and Logic: Answer Set Programming and SAT*, pages 19–34, 2006.
13. Aleecia M McDonald and Lorrie Faith Cranor. The cost of reading privacy policies. *Information System: A journal of law and policy for the information society*, 4:543–897, 2008.
14. Michael K Reiter and Aviel D Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security (TISSEC)*, 1(1):66–92, 1998.
15. Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In *Privacy Enhancing Technologies*, pages 41–53. Springer, 2003.
16. US Department of Health and Human Services and others. Summary of the HIPAA privacy rule. *Washington, DC: Department of Health and Human Services*, 2003.
17. Meilof Veeningen, Benne de Weger, and Nicola Zannone. Formal privacy analysis of communication protocols for identity management. In *Information Systems Security*, pages 235–249. Springer, 2011.

Appendix A Details on the Prolog implementation of the first use case

The use case-specific input are definitions of relevant credentials, the providers' policies, the specification of the unique attributes and attribute sets and the user's specific requests for services' utilisation. The definitions of credentials (such as C1 and C2, cf. further) should be provided by the credential issuer, rather than requiring the users to create these models themselves. The policies of the providers that define how user data is managed should also be provided by the providers themselves. This limits the modelling effort that is performed on the user side significantly.

The user's credentials that are defined in the observed use case are the Belgian national ID credential and the user's student card:

```
%Credential definition C1 : BeID :
:-defineCredType('X.509', 'BeID',
        [attrDef('Name', 'String', 'Known', 'Assigned', 'Shared'),
        attrDef('Surname', 'String', 'Known', 'Assigned', 'Shared'),
        attrDef('Address', 'String', 'Known', 'Assigned', 'Shared(limited)'),
        attrDef('DoB', 'Date', 'Known', 'Assigned', 'Shared'),
        attrDef('PK', 'PublicKey', 'Chosen', 'Known', 'Unique')]).
```

```
%Credential definition C2 : Student card :
:-defineCredType('X.509', 'studentcard',
        [attrDef('Name', 'String', 'Known', 'Assigned', 'Shared'),
        attrDef('Surname', 'String', 'Known', 'Assigned', 'Shared'),
        attrDef('Stud#', 'Integer', 'Known', 'Assigned', 'Unique'),
        attrDef('University', 'String', 'Known', 'Assigned', 'Shared'),
        attrDef('PK', 'PublicKey', 'Chosen', 'Known', 'Unique')]).
```

The unique attributes are defined with:

```
:-assert(isUnique('PK', credSource('X.509', 'stud. card', 'KULeuven'))).
:-assert(isUnique('PK', credSource('X.509', 'BeID', 'gov'))).
```

Attributes that are considered to create a unique set are also defined:

```
:−assert(uniqueSet(
        [attrNameSourcePair('Stud#',
            credSource('X.509', 'studentcard', 'KULeuven')),
        attrNameSourcePair('DoB',
            credSource('X.509', 'BeID', 'gov'))])).

:−assert(uniqueSet(
        [attrNameSourcePair('StudentNumber',
            credSource('X.509', 'stud. card', 'KULeuven')),
        attrNameSourcePair('University',
            credSource('X.509', 'stud. card', 'KULeuven'))])).
```

The models of unique attributes and attribute sets should be provided by issuers, who have an overview of the complete system. However, in case of privacy-concerned users, they may add their own rules that create a 'stricter' model to evaluate the information that is known about them.

The authentication policy of the provider is another part of the input. Its model should be provided by the service provider itself, rather than requiring their implementation on the user side. A policy model that is considered in the example is:

```
%Authentication policy A1 : general purchase at OnlineShop :
:−assert(authPolicy('OnlineShop', 'purchaseGeneral',
    [attrAndSource(cred(_, 'X.509', 'BeID', _), ['DoB']),
    attrAndSource(cred(_, 'X.509', 'stud. card', _),['Ownership']),
    attrAndSource('userInput', ['purchase'])])).
```

Final input are the user's requests to utilise the purchasing service:

```
:−reqService('Alice', 'OnlineShop', '2', 'purchaseGeneralProducts').
:−reqService('Alice', 'OnlineShop', '3', 'purchaseGeneralProducts').
```

For every transaction, the user needs to provide only one such line of code. Ideally, this would be performed automatically by a supporting tool that would model every service request that a user makes. In the same way, it could model transactions that a user is considering of making to provide feedback on them.

Based on the authentication policy that is defined, the disclosures are determined. The created object code of the presented modelling can be queried about the links that different providers are able to establish. An example is checking which links a specific party is able to establish:

```
?− isLinked(Provider, TID1, TID2, Reason).
```

Given the initial state of the knowledge database and the disclosures that the user makes, a snippet of the obtained result has the form:

```
Provider = 'OnlineShop',
TID1 = '2',
TID2 = '3',
Reason = reason('uniqueSet',
    [attrNameSourcePair('Stud#', credSource('X.509', 'stud. card',
    'KULeuven')),
    attrNameSourcePair('DoB', credSource('X.509', 'BeID', 'gov'))]).
```